# Congestion Control in Reliable CoAP Communication

August Betzler, Carles Gomez, Ilker Demirkol, Josep Paradells
Department of Telematics Engineering
Universitat Politecnica de Catalunya
i2CAT Foundation
Barcelona, Spain
{august.betzler, carlesgo, ilker.demirkol, josep.paradells}@entel.upc.edu

## ABSTRACT

The development of IPv6 stacks for wireless constrained devices that have limited hardware resources has paved the way for many new areas of applications and protocols. The Constrained Application Protocol (CoAP) has been designed by the IETF to enable the manipulation of resources for constrained devices that are capable of connecting to the Internet. Due to the limited radio channel capacities and hardware resources, congestion is a common phenomenon in networks of constrained devices. CoAP implements a basic congestion control mechanism for the transmission of reliable messages. Alternative CoAP congestion control approaches are a recent topic of interest in the IETF CoRE Working Group. New Internet-Drafts discuss the limitations of the default congestion control mechanisms and propose alternative ones, yet, there have been no studies in the literature that compare the original approach to the alternative ones. In this paper, we target this crucial study and perform evaluations that show how the default and alternative congestion control mechanisms compare to each other. We use the Cooja simulation environment, which is part of the Contiki development toolset, to simulate CoAP within a complete protocol stack that uses IETF protocols for constrained networks. Through simulations of different network topologies and varying traffic loads, we demonstrate how the advanced mechanisms proposed in the drafts perform relative to the basic congestion control mechanism.

## Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless Communication; C.2.2 [**Network Protocols**]: Applications, Protocol Architecture

## General Terms

Algorithms, Performance, Experimentation, Verification

## Keywords

CoAP; RPL; 6LoWPAN; IEEE 802.15.4; Cooja; Contiki; End-to-end Reliability; Congestion Control

## 1. INTRODUCTION

The challenge of designing protocols and standards for constrained devices has become greater, as the networks they form are no longer only of local character but also get connected to the Internet as part of the Internet of Things [12]. The Internet Engineering Task Force (IETF) has been focusing on finding solutions for different aspects of the communication protocol stack that are trimmed to the requirements of networks of constrained devices. The IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [15], IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) [8] and CoAP [13] are some of the protocols designed by the IETF working groups in this space.

The limited hardware capacities of sensor nodes and the limited link capacities of the radio channel may lead to congestion in constrained networks. Congestion may result in increased delays, because packets remain longer in node buffers before being successfully transmitted over wireless links or it may lead to packet losses, when internal message buffers overflow or the simultaneous transmission of packets in the radio channel leads to collisions that render packets useless.

Congestion control mechanisms are able to detect congestion and/or apply measures to avoid or dilute congestion, which can have an important effect on the network performance. Thus, the investigation of congestion control mechanisms in the ambit of constrained networks that run CoAP is also crucial. As an alternative to the basic mechanisms applied by CoAP, the CoAP Simple Congestion Control/Advanced (CoCoA) draft [1] proposes more complex but potentially improved congestion control mechanisms for CoAP. However, performance evaluations are needed to confirm if the proposed mechanisms result in a more efficient congestion control. In a novel approach, we use the Cooja simulation environment for Contiki, an operating system employed by many constrained devices, to carry out simulations that determine the performance of the alternative CoAP congestion control mechanisms for different network topologies and varying traffic loads. We evaluate these mechanisms over a stack of protocols which includes protocols specified by the IETF for networks of constrained devices. To the best of our knowledge, this is the first work to evaluate CoCoA performance and furthermore the first study that evaluates congestion control for CoAP.

The rest of the paper is organized as follows. In Section 2, the basic CoAP congestion control mechanism is explained along with the alternative CoCoA mechanisms. In Section 3, the communication protocol stack used for the evaluation of the CoAP congestion control performance is detailed. Also, the simulation parameters and network setups are explained. The results of the simulations are presented in Section 4. Section 5 concludes the paper, giving a resume of the obtained results and giving indications for further analysis on CoAP congestion control for constrained networks.

## 2. COAP CONGESTION CONTROL

CoAP is a Representational State Transfer style (RESTful) [5] protocol developed for constrained devices that allows interactions between clients and servers over the Internet. In the environment of constrained devices, CoAP can be used for the manipulation of resources, such as sensor measurements or actuator states.

For simplicity, CoAP uses UDP as the transport protocol by default. Since UDP does not implement end-to-end reliability but this may be required by the application, CoAP introduces optional application layer end-to-end Acknowledgements (ACKs). By setting the confirmable (CON) flag in an outgoing CoAP message, an end-to-end ACK is requested from the destination node. If no ACK is received within the retransmission timeout (RTO) interval, a retransmission is initiated for the outstanding transaction. The initial RTO for any CoAP transaction as specified by the base CoAP specification [13] is randomly chosen from the fixed interval [2s, 3s].

A relatively high initial value is chosen, because packets that travel through the Internet may suffer large delays and round-trip times (RTTs). Apart from that, a CoAP request may trigger an action at the destination device that requires intensive processing, which can increase the RTT due to processing delay.

If a message is not confirmed during the RTO interval, CoAP assumes that its transmission has failed. However, the cause for an unsuccessful confirmation is not explicit, since the CoAP message or its corresponding ACK may have been delayed beyond the duration of the RTO interval or they may have been lost, because of congestion or due to lossy wireless links. Because of this ambiguity it is difficult to determine an optimal RTO. On one hand, it should not be too short, to avoid spurious retransmissions in the case that confirmation of the request suffers from delay. On the other hand, it should not be too long, to avoid the originator of the CoAP request to idle unnecessarily before retransmitting the packet in the case that the request or confirmation packets were lost. In this context, the basic RTO calculation of CoAP (a fix interval of possible values) is quite trivial and inflexible. It may not provide an optimal performance for all types of scenarios.

Apart from retransmitting the message, once the RTO timer expires, CoAP applies a binary exponential backoff (BEB) as in the Transmission Control Protocol (TCP), effectively throttling the generation of more packets, as it assumes that the network is congested. CoCoA, amongst other things, proposes to run two RTO estimator algorithms for each destination of CoAP transactions, the so called strong and weak estimators. The strong estimator uses the RTT that is measured if an ACK was received for a transaction that did not require retransmissions. The weak estimator uses the RTT that is measured when an ACK is received after retransmitting the CoAP request at least once. Both RTO estimators use the RTO formulas introduced in RFC 6298 [11]. When the overall RTO is updated, a weighted average of the previous overall RTO and the last obtained RTO estimation, being a weak or strong one, is calculated. The overall RTO is initialized to 2 seconds in CoCoA.

The evaluation of the CoAP congestion control performance in this paper includes the evaluation of the default and two alternative RTO calculations. The first alternative is the RTO calculation as proposed in CoCoA, which requires additional state information and program logic. As a lightweight alternative we evaluate a second congestion control algorithm, CoCoA Strong (CoCoA-S), which only considers the strong estimator.

Another important CoAP parameter that is discussed in CoCoA is the NSTART parameter, which determines how many CoAP transactions may be created to one destination endpoint in parallel. The base CoAP document states that this value must be set to 1. Higher values can lead to congestion of the network, since it allows various CoAP transactions between two nodes simultaneously. How the performance of the default and advanced congestion control changes when using a NSTART value of 4 is also analyzed in the experiments presented in Section 4. For the case of non-confirmable messages, CoCoA proposes some restrictions, however in this paper we only focus on reliable communication.

## 3. SIMULATION SETUP

This section introduces the Cooja simulator and gives an overview of the Contiki communication protocol stack used in this study. Also, this section covers the configuration of the nodes and explains the tested network topologies, as well as the test run configuration used to do the performance evaluations.

### 3.1 Contiki OS and the Cooja Simulator

Contiki OS is an open source operating system designed for constrained devices and the Internet of Things and is the choice of many sensor motes. Fig. 1 shows the IETF protocols over IEEE 802.15.4 and how they are implemented in Contiki OS, along with other Contiki specific layers. It implements the uIPv6 stack that provides the functionality for IPv6 networking. uIPv6 includes Contiki-RPL, an implementation of the routing protocol for low-power lossy IPv6 networks, and SICSlowpan, an implementation of the 6LoWPAN adaptation layer. Contiki also comes with the Erbium CoAP implementation [7], which currently supports CoAP up to draft version 13, together with observe and blockwise transfers [6, 2]. In the layer based model of the communication protocol stack, CoAP is located at the application layer on top of uIPv6.

RPL, as part of the uIPv6 Stack, is responsible for the network formation, packet routing and the maintenance of the network topology. In RPL, one or more root nodes generate a network topology that trickles down to the leaf nodes, resulting in Directed Acyclic Graphs (DAGs). If the network contains a single root, it is called Destination Oriented DAG (DODAG). Contiki 2.6 only supports the storing mode for RPL, where each node stores routing entries that indicate the next hop(s) to the RPL-root and to its child nodes. The RPL objective function (OF) determines which links are

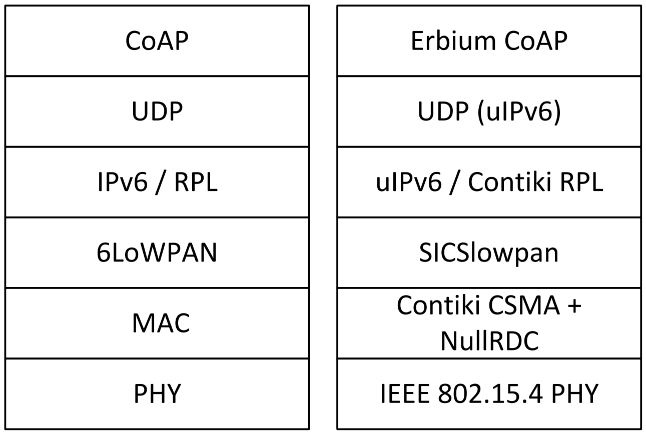


Figure 1: A comparison between the IETF communication protocol stack (left) and its implementation in Contiki (right).

chosen to be part of the RPL-DAG. Contiki comes with two predetermined OFs, the expected transmission count (ETX) OF [4] is the default OF and is used in the evaluation carried out in this paper.

The 6LoWPAN implementation of the uIPv6 stack, SICSlowpan, applies header compression of IPv6 packets at the sender site, and fragmentation if necessary to transport IPv6 packets on top of IEEE 802.15.4 MAC layer frames. At the receiver side, SICSlowpan takes care of decompressing the headers to reestablish their original content and performing reassembly if needed. Underneath the layers that are part of uIPv6, Contiki implements a MAC driver with Carrier Sense Multiple Access (CSMA) and link layer reliability. Contiki allows choosing from different radio duty cycling (RDC) drivers that determine when a mote turns off the radio to save energy.

Cooja is a network simulator that can execute Contiki OS-based code, designed to be a rapid development platform for Contiki. Cooja supports simulation of sensor networks at three levels: the application level, the operating system, and the machine code instruction set [10]. As an outstanding feature, the binary image of compiled Contiki code designed for real motes can be uploaded to be used for simulations. Cooja also is capable of emulating the real hardware for several types of nodes. All together a high degree of simulation realism is achieved, since memory restrictions, the real node periphery and internal processing of the motes are simulated as well.

Cooja allows three-dimensional deploying of nodes inside the simulated environment and offers several radio models. The radio models determine how radio signals propagate in the network. The radio channel model used is the unit disk graph radio model, which defines the link delivery ratio within the transmission range to be 100%. The interference range, set to be twice the transmission range, indicates up to which distance the radio signal is capable of interfering the radio of other nodes. If a node receives two packets at the same time (being in transmission or interference range), the simulator assumes a packet collision, rendering both packets useless for the receiving node. More complex models like a ray-tracing based model are under development or not yet fully implemented.

## 3.2 Sensor Mote Configuratio

Cooja provides the necessary software to emulate the hardware of a set of real sensor nodes. This means that real hardware limitations like memory and processing capacities are respected in the simulation environment. From the types of devices that are supported by Cooja, two IEEE 802.15.4 capable motes with the same type of radio transceiver (CC2420 [14]) are chosen for the simulations: the Z1 from Zolertia [16] and the Tmote Sky from Moteiv [9], latter being equivalent to TelosB motes [3] in terms of hardware. The most relevant features of the two motes are stated in Table 1. As seen in the table, the motes differ in ROM and RAM capabilities and the MCU. While Z1s offer more ROM for application code, the Sky Motes have higher RAM storage capacity.

Table 1: Hardware Specifications of the Zolertia Z1 and Moteiv Tmote Sky Wireless Sensor Nodes.

| | Tmote Sky | Z1 |
|---|---|---|
| RAM | 10 KB | 8 KB |
| ROM | 48 KB | 92 KB |
| MCU | MSP430F1611 | MSP430F2617 |
| Radio | CC2420 | |

The simulated network topologies used for CoAP performance evaluation include two types of nodes that differ in their resource requirements: CoAP nodes and RPL border routers. Each topology includes one RPL border router that implements the communication stack introduced in Section 3.1 up to UDP and that acts as RPL-root node. The simulated network topologies are local networks that do not require the border router to establish external connections.

According to the RPL specifications, when a node is missing a routing table entry for the destination node, it forwards the packet to the next node in direction of the RPL-root. This process is repeated, until a node has a valid routing entry to forward the packet in direction of the requested destination address. In the case that none of the nodes on the way to the RPL-root has valid routing information stored, the RPL-root eventually receives the packet. The RPL-root then in the optimal case looks up the entry with the destination address for the packet from the routing table and forwards it to the next hop in direction of the destination node. To achieve that, the RPL border router needs to be able to store routing table entries for all possible destinations in the network, which is why the RAM requirement of this node is relatively high. Because of that, the RPL border router runs on Tmote Sky nodes that have a higher RAM capacity compared to the Z1 motes. Since the analyzed network topologies are of limited size, it is possible to assign sufficient routing table sizes to the RPL-root beforehand for each topology.

The rest of the motes in the network topologies are nodes that run the full Contiki OS stack and the Erbium CoAP implementation based on CoAP version 13. In contrast to the RPL-root, the nodes with CoAP require more ROM to fit the additional application code that is part of the Erbium CoAP implementation and also for the user application that controls the high level behavior of the node. Z1 nodes are better suited for this type of tasks, since they have higher ROM capacities when compared to Tmote Sky nodes.

The two types of motes are part of all network topologies that are used for simulation experiments in this paper. In

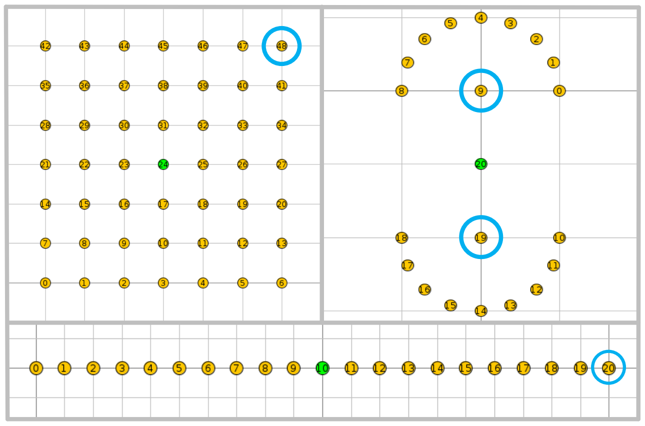


Figure 2: The three network topologies used for performance analysis (grid, dumbbell, and chain). The width of a square corresponds to 10 m. Nodes with a blue circle are sink nodes for CoAP messages. Green nodes are RPL border routers.

each of the network topologies, there is always a single RPL-root, while the rest are CoAP nodes, out of which one or two are set as the destination (sink) nodes as detailed in Section 3.3.

Another factor that influences the performance of the network essentially is the size of Contiki's MAC layer queue and the RDC mode. All packets that are generated or forwarded by a node need to traverse a queue located at Contiki's MAC layer. The queue is used to store packets that are pending to be sent and that need to wait the CSMA procedure before being transmitted by the radio. Since the amount of RAM for constrained devices is very restricted, the number of MAC layer queue entries is set to 4 for CoAP nodes and 6 for the RPL border router. Contiki disposes of different RDC mechanisms, such as NullRDC, B-MAC, X-MAC and ContikiMAC. For the evaluations carried out in this paper, the NullRDC is chosen. In NullRDC, nodes do not sleep and their radio is constantly turned on. For the performance analysis of CoAP, we discard to use any RDC alternative, thus avoiding the cross-layer interactions they require.

## 3.3 Network Topologies

To carry out the performance evaluation of CoAP congestion control mechanisms, three different network topologies are defined for the simulations. They differ in the number and positioning of the nodes, which amongst other things lead to differences in the number of direct neighbors of the nodes, the lengths of routes between source and destination of CoAP requests, and the number of nodes that may compete for the radio channel simultaneously. The evaluated topologies are i) a chain of nodes with 21 nodes, ii) a rectangular grid of 49 nodes (7x7), and iii) a dumbbell topology with 21 nodes.

Figure 2 depicts snapshots of the Cooja simulator GUI showing the two-dimensional positioning of the nodes for the three different topologies. The transmission range is set to 10 meters, whereas the interference range is set to be twice of the transmission range, i.e., 20 meters. The distance between nodes in the chain topology is chosen in such a way that only direct neighbors are in transmission range. Since the interference range is twice the transmission range, a packet transmission may interfere the radio of nodes up to two hops away. Each node may therefore have up to two neighbors. The same distance rule applies for the grid topology, where each node may have up to 4 direct neighbors. In the dumbbell topology, the nodes on the half circles are exactly 10 meters away from the node that is part of the dumbbell's axis.

To observe the performance of the CoAP congestion control algorithms, we want to avoid that RPL malfunctioning distorts the results. The network topology determines the configuration for some of the RPL parameters, that, when not configured adequately, may lead to performance drops. The routing table and neighbor table sizes are two important parameters that determine if packets can be routed correctly throughout the network. For the simulations carried out in this paper, the neighbor tables are able to store information about all direct neighbors. The RPL border route has a routing table large enough to store entries for all nodes of the network. Due to the strict RAM limitations of CoAP nodes, their routing tables need to be smaller. Adjusted to the introduced topologies, the storable amount of RPL neighbor and routing table entries is set to the values shown in Table 2.

Table 2: Configurations of the RPL Routing and Neighbor Table Sizes.

| | Chain Topology | Dumbbell Topology | Grid Topology |
|---|---|---|---|
| Neighbor Table (all nodes) | 2 | 10 | 4 |
| Routing Table (CoAP node) | 10 | 10 | 20 |
| Routing Table (RPL bord. router) | 20 | 20 | 48 |

Since the RPL border router acts as dispatcher for messages that need to be forwarded through the RPL root, it should be located in a central position of the network to make it equally accessible from all nodes in the network. In the chain topology, the RPL border router is located in the middle of the chain. In the grid it is located at the center of the topology, where the 4th line of nodes crosses with the 4th column. In the dumbbell topology the RPL border router is the center node that connects the two laterals of the dumbbell formation. In all the topologies, CoAP messages are created by the CoAP nodes and directed to one or two sink nodes. Except for the dumbbell topology, each network has a single sink, marked with a circle in the topology overviews. In the dumbbell topology there are two sink nodes. They are the destinations of traffic that is generated by nodes from the opposite side of the dumbbell topology. While in the chain and grid topologies the sink nodes do not generate any CoAP requests, in the dumbbell topology each of the sink nodes also creates CoAP requests for the sink node on the opposite side.

## 3.4 Test Run Configuratio

When a test run is started, the simulated nodes are initialized and the RPL-root creates the RPL-DODAG by spreading DAG Information Objects (DIOs) throughout the network. CoAP performance tests begin after the complete RPL-DAG is built. After this, the nodes start generating periodical CoAP requests. The generation of periodic traffic is implemented by running cyclic timers that upon their depletion create a new CoAP request. The CoAP request is a POST message that sets the color of a LED at the sink node and increases an internal counter of the sink node by one. Additionally an increasing message ID and the short address (node ID) of the originator node are included in the payload of the CoAP request. The sink node that receives this CoAP request carries out the requested actions and responds to the originator node with a confirmation message. A CoAP request message including all headers and the payload has a size of 95 bytes.

If a node generates a CoAP request, but the limitation of parallel transaction to the destination node dictated by NSTART does not permit to generate another request, the generated packet is dropped. Since all CoAP requests in the analyzed network topologies only have one possible destination, packets at the application layer are only dropped when a node is waiting for a confirmation of the previously sent CoAP request in the case of NSTART=1. If NSTART is larger, packets at the application layer are only dropped, if already NSTART transactions are active. In real life scenarios, where a single sink node is employed, NSTART can play an important role for possible congestion of the network. Thus, we also analyze the performance of congestion control mechanisms for CoAP with a higher NSTART value.

The duration of the simulation phase during which CoAP requests are generated is always 360 seconds. After this time, the test script generates a signal for all nodes in the network that tells them to stop spawning further CoAP requests. Subsequently the simulation terminates. All events that occur during the simulation and are of interest for the performance evaluation are collected from the nodes over their virtual serial interface. During the simulation, the nodes can send messages over their serial port hat can be intercepted by the simulation script and can then be stored into log-files for further evaluation. Such messages can contain the status of internal variables, such as buffer sizes or can be used to notify about an event, such as the reception of a confirmation message.

In the following section, the performance results of the investigated approaches are presented.

## 4. SIMULATION RESULTS

This section presents the results of the experimental evaluation of the congestion control mechanisms in reliable CoAP communication for the three previously introduced network topologies. As the performance metric, the relation of carried load against the offered load is chosen. The offered load refers to the total amount of data created by the nodes per second on average when CoAP requests are generated, and is given in kilobits per second (kbps). The carried load indicates the average amount of data that is successfully delivered to the sink node(s) per second, and also given in kbps.

For all analyzed configurations of the CoAP congestion control mechanisms, the test runs have been repeated three times with different random simulator seeds to obtain meaningful average results.

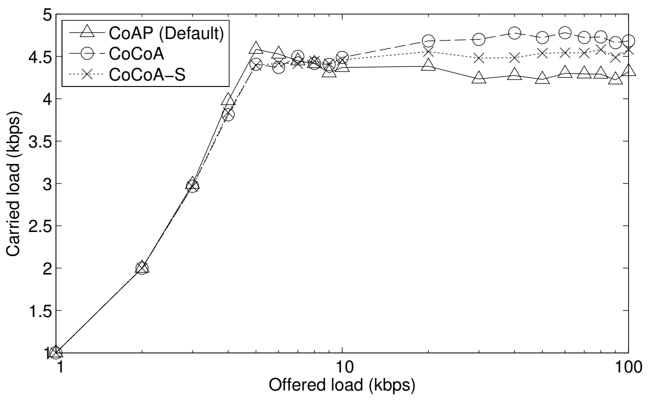


Figure 3: Throughput achieved for different offered loads in the dumbbell topology with NSTART=1.

### 4.1 Congestion Control Performance for Different Traffi Loads

When NSTART is set to 1, the main difference of the analyzed congestion control algorithms lies in the RTO calculation algorithm. The results show that this mechanism has a high impact on the network performance and that it influences the effectiveness of the CoAP congestion control.

Fig. 3 illustrates how the carried load evolves as the offered load increases when using the three RTO algorithms for the dumbbell topology evaluated. Since at low traffic rates, the network does not reach a state of congestion and nearly all packet transmissions across the network are successful, the carried load is identical to the offered load, independently of the RTO algorithm used. As the offered load increases, the ratio of successfully delivered CoAP requests decreases because of congestion. This is where differences in the performance of the different congestion control algorithms become visible.

The performance of the three congestion control algorithms is almost identical up to an offered load of 3 kbps. For offered traffic loads up to 7 kbps, CoCoA and CoCoA-S show a slightly worse performance in terms of throughput. Since the number of hops between source and destination routes is small and there is almost no congestion in form of packet collisions or packet drops, the CoCoA algorithms have small RTO estimations. As the estimated RTO gets smaller and reaches the real RTT, the probability for spurious retransmission increases and sometimes causes the nodes to unnecessarily retransmit a packet they assumed to have been lost. These spurious retransmissions can lead to drops in the throughput performance. The default CoAP RTO does not suffer from these effects, as the initial RTO for transactions always is 2 s or higher. As the offered traffic load increases beyond 7 kbps, the congestion control mechanisms of CoCoA and CoCoA-S take effect, outperforming default CoAP. CoCoA-S has a slightly lower performance than CoCoA. At high traffic rates that lead to increased packet drop rates across the network, CoCoA-S has a lower probability to obtain valid RTT measurements, thus it is not able to calculate improved RTO estimations.

The relevance of RTO estimations becomes clearer when looking at the amount of packets that are dropped during the simulation. Packet drops occur when i) the MAC layer buffer overflows, or ii) when CoAP refuses to send a packet, because the number of parallel transactions allowed for a

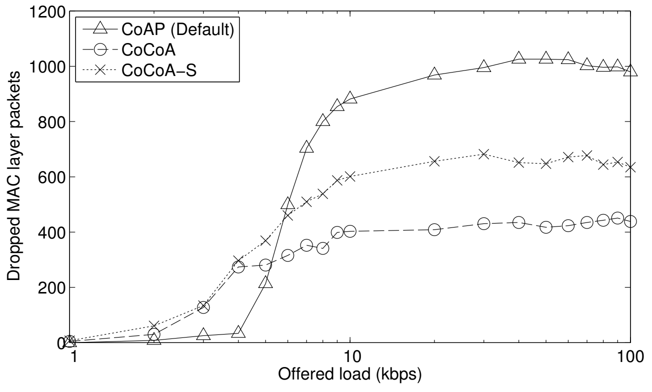


Figure 4: Dropped MAC layer packets for different offered loads in the dumbbell topology with NSTART=1.

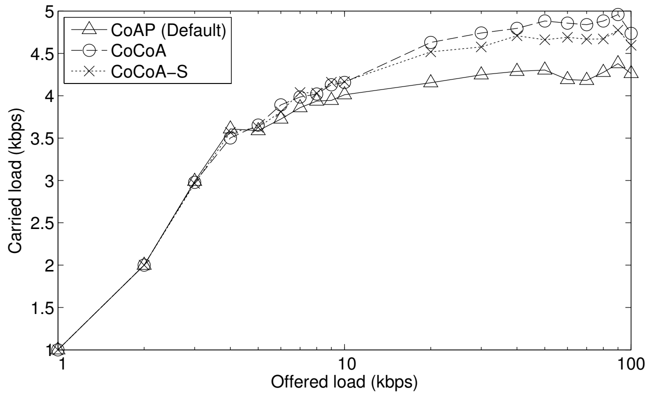


Figure 5: Throughput achieved for different offered loads in the chain topology with NSTART=1.

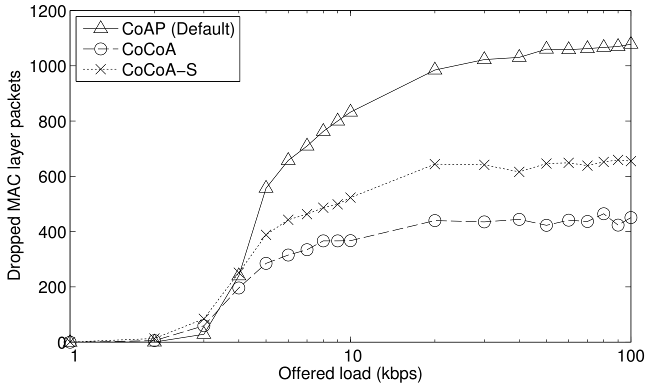


Figure 6: Dropped MAC layer packets for different offered loads in the chain topology with NSTART=1.

destination, i.e., NSTART parallel transactions, has been reached for the destination of the packet.

MAC layer buffer overflows are a clear sign of congestion, as they indicate that a high amount of packets are created and intended to be forwarded throughout the network, exceeding the capabilities of the network. The effect of the RTO algorithm on the amount of dropped packets is deducted for the dumbbell topology and Fig. 4 shows the overall amount of dropped MAC layer packets for this topology.

As seen in the figure, when applying the default RTO algorithm, the number of MAC layer packet drops grows quickly above 3 kbps of offered traffic load and reaches an asymptotic value at around 20 kbps. When using the alternative RTO algorithms for CoAP congestion control, a slower increment of the number of dropped packets and a much lower asymptotic value are observed. This means that the CoCoA algorithms are capable of effectively decreasing the congestion in the network. The clear difference in the number of dropped MAC layer packets between CoCoA and CoCoA-S is a result of the different RTO estimations they apply. In contrast to CoCoA-S, CoCoA additionally uses weak RTT measurements from retransmissions that may lead to large RTOs. This throttles the output of packets and leads to a further reduction of buffer overflows in the network.

An asymptotic behavior of the amount of dropped MAC layer packets is observable, because the actual amount of traffic transmitted over the radio channel does not increase significantly further with the offered load at high traffic rates. On the other hand, due to the limitation given by NSTART=1 and increasing packet generation frequencies at each node, the probability of dropping newly generated CoAP messages gets higher with the offered load. For offered load beyond the threshold where the amount of dropped MAC layer packets reaches its asymptotic value, the CoAP packet drop rate is observed to increase almost linearly with the amount of generated packets.

As mentioned before, the amount of hops between source and sink nodes in the dumbbell topology is low, resulting in small RTTs. How the congestion control algorithms perform when larger RTTs are observed can be demonstrated in the chain topology, where packets may have to travel along many hops before reaching their destination. The radio links along the chain are not utilized equally. The links closer to the sink will be required more frequently for transmissions, as all the packets from the other end of the chain need to traverse them. On the other hand, due to spatial reuse, it is possible for several nodes to transmit in parallel along the chain, without interfering each other.

For this particular setup, the difference between the default CoAP and the CoCoA mechanisms becomes larger, as can be seen in Fig. 5. This shows that the CoCoA mechanisms are able to adapt to different RTT values and to the traffic characteristics of the network. This observation can be backed up by observing the amount of dropped MAC layer packets (Fig. 6), showing a similar behavior as in the dumbbell topology, where the drop rates for the CoCoA mechanisms are much smaller than for default CoAP.

On the other hand, in the grid topology, the number of hops between source and destination nodes varies a lot, since many combinations of links for the connection of source and sink nodes are possible. Therefore, RTTs of different scales can be observed. Since within the transmission, and also the interference range of a node there will be more nodes, a congestion of the radio channel is more likely than in the other analyzed topologies.

The performance comparison for the grid topology reveals that CoCoA mechanisms are able to perform significantly better than default CoAP. Fig. 7 shows that CoCoA and CoCoA-S perform better than the default CoAP, even when the offered load is small (starting at 4 kbps). However, the performance achieved by the two advanced congestion control mechanisms is very similar for this topology. For the intermediate traffic rates, CoCoA gets many weak RTT measurements from retransmissions, increasing the es-

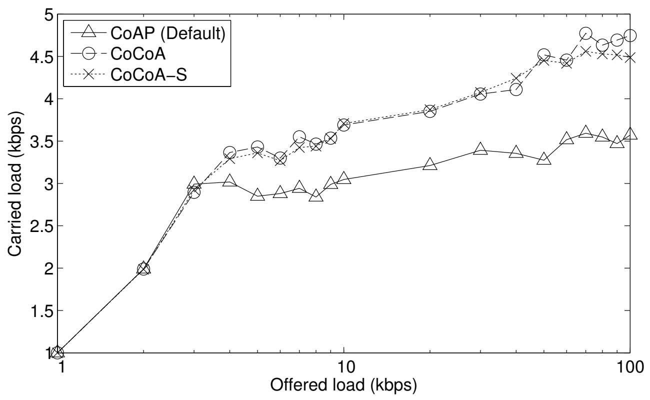


Figure 7: Throughput achieved for different offered loads in the grid topology with NSTART=1.

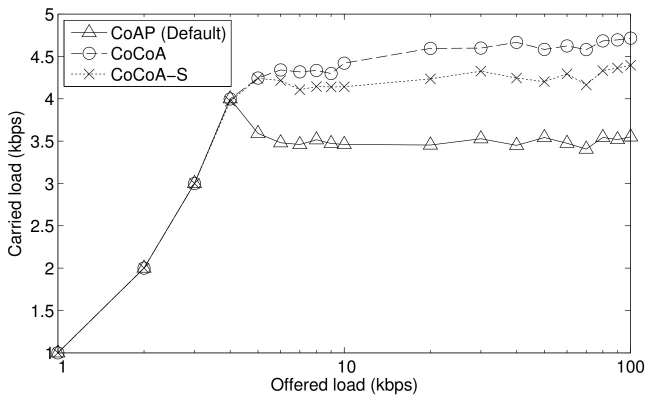


Figure 8: Throughput achieved for different offered loads in the dumbbell topology with NSTART=4.

timated RTO to large values. In case of a loss this results in a long idle period. CoCoA-S and default CoAP use shorter RTOs. The amount of valid RTT measurements gets lower with higher traffic for nodes that apply CoCoA-S and that are not very close to the sink. Thus the CoCoA-S RTO estimations do not grow as much as CoCoA RTO estimations, while default CoAP maintains the default initial interval. Due to these lower RTO estimations, CoCoA-S will use more retransmissions in a fixed time interval for nodes that have difficulties to successfully complete transactions on their first transmission, when compared to CoCoA. This can increase the PDR, until the traffic rates become higher and this behavior is the cause of more congestion. This is the case above 60 kbps, where the performance of CoCoA-S is lower than the performance of CoCoA. The results for the MAC layer packet drops are very similar to the results for the chain and dumbbell topology and will not be detailed further. As before, CoCoA and CoCoA-S lead to a much lower drop rate of MAC layer packets.

The applicability of the observations to several network topologies confirms that the improvements obtained by using CoCoA and CoCoA-S are crucial for reliable CoAP communication. In the following, this conclusion is strengthened by observing the performance of the congestion control mechanisms when allowing multiple parallel CoAP transactions to a destination node (NSTART=4).

### 4.2 Effect of NSTART on Congestion Control Performance

When NSTART is set to 4, the amount of maximum parallel transactions to one destination is increased from 1 to 4. Nodes that apply the basic CoAP congestion control initialize independent RTO timers for each transaction. Nodes that create transactions when using CoCoA and CoCoA-S can resort to already obtained RTO information for a destination node.

With a higher NSTART value, the potential degree of congestion in the network is higher, increasing the importance of an efficient congestion control mechanism.

Fig. 8 illustrates the carried load in the dumbbell topology. As seen in the figure, the default congestion control mechanism is not able to cope with NSTART=4. When using the default CoAP congestion control mechanism the performance drops noticeably compared to the performance achieved with NSTART=1. For default CoAP, offered load above 4 kbps leads to a relevant drop and the carried load starts fluctuating around a value of 3.5 kbps. Analogical to the NSTART=1 scenario, CoCoA and CoCoA-S are able to surpass the performance of the default CoAP congestion control. However, the differences in the performance are greater for NSTART=4. When analyzing the amount of dropped MAC layer packets (Fig. 9a), it becomes clear that the default CoAP is not able to control several parallel transactions efficiently. The number of MAC layer packet drops is 3 to 4 times higher, compared to the number of drops observed for CoCoA and CoCoA-S. This means that the advanced congestion control mechanisms are able to better detect congestion and reduce it.

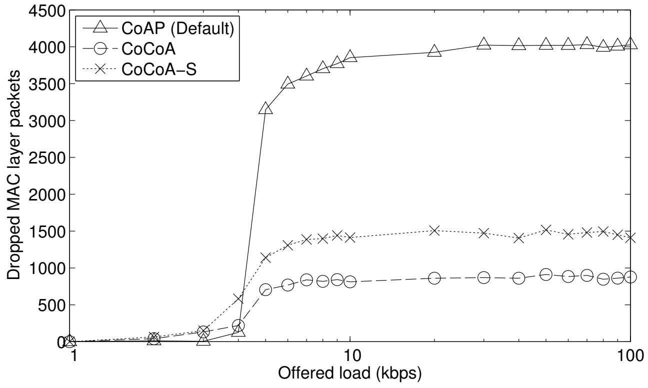
Figs. 9b and 9c show the throughput results for the chain and grid topology, respectively. The observations for NSTART=1 also apply for NSTART=4, but with a greater difference between the three congestion control mechanisms in the dumbbell and chain topologies. Due to space limitations, the figures for the number of MAC layer drops are omitted.

## 5. CONCLUSIONS

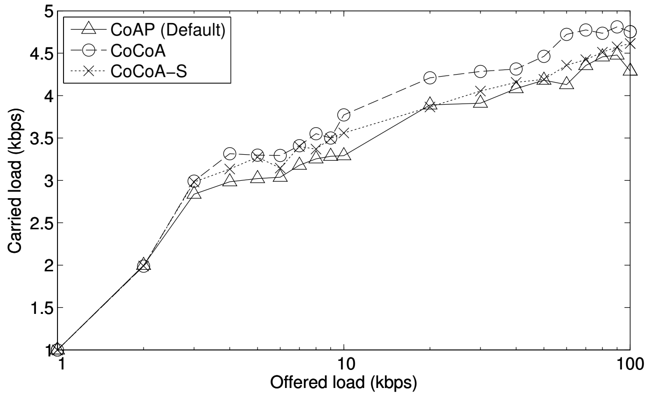
In this paper, we have carried out a performance evaluation of different CoAP congestion control mechanisms proposed for reliable end-to-end communication. For that, we used the Cooja simulation environment to evaluate CoAP within the Contiki communication protocol stack that includes IETF protocols for constrained networks. Three different network topologies are simulated with different network traffic loads.

We showed that the recently proposed CoCoA congestion control mechanism for CoAP on average performs equal to or better than the default congestion control mechanism in terms of throughput. CoCoA is able to reduce the quantity of MAC layer buffer overflows for congested networks, an important aspect in constrained networks. The results obtained with NSTART set to 1 could be confirmed for the case where NSTART is set to 4, where an even greater difference between the performance of the default CoAP congestion control and CoCoA was observed.

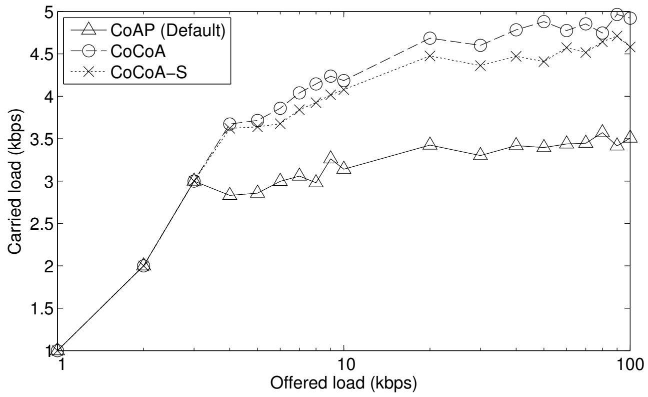
We also evaluated a simplified version of CoCoA, namely CoCoA-S. The simplified CoCoA-S mechanism does not perform as well as the CoCoA mechanism, accentuating the need to use the weak RTO estimator, as opposed to using only the strong RTO estimator. However, CoCoA-S is able



(a) Dropped MAC layer packets for different offered loads in the dumbbell topology with NSTART=4.



(b) Throughput achieved for different offered loads in the grid topology with NSTART=4.



(c) Throughput achieved for different offered loads in the chain topology with NSTART=4.

Figure 9: Congestion control performances for NSTART=4.

to outperform the default CoAP congestion control mechanism, while not requiring as much state information as CoCoA. With the results obtained in this paper, we can confirm the relevance of advanced congestion control mechanisms and confirm that for this purpose CoCoA is an interesting candidate.

As future work, the advanced congestion control mechanisms will be tested in more network topologies, other performance metrics will be evaluated and different traffic generation patterns will be applied.

## 6. ACKNOWLEDGMENTS

This work has been supported by the Spanish Government's Ministerio de Economía y Competitividad through project TEC2012-32531 and by MICINN through project TEC2009-11453, as well as by the Comissionat per a Universitats i Recerca del DIUE from the Generalitat de Catalunya, the Social European Budget (Fons Social Europeu), and FEDER.

## 7. REFERENCES

[1] C. Bormann. CoAP Simple Congestion Control/Advanced (work in progress), August 2012.

[2] C. Bormann and Z. Shelby. Blockwise transfers in CoAP (work in progress), Feb. 2013.

[3] Crossbow Technology Inc. TelosB mote platform, 2009.

[4] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, MobiCom '03, pages 134–146, New York, NY, USA, 2003. ACM.

[5] R. T. Fielding. *Architectural styles and the design of network-based software architectures.* PhD thesis, 2000. AAI9980887.

[6] K. Hartke. Observing Resources in CoAP (work in progress), Feb. 2013.

[7] M. Kovatsch, S. Duquennoy, and A. Dunkels. A low-power coap for contiki. In *Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2011)*, Valencia, Spain, Oct. 2011.

[8] N. Kushalnagar, G. Montenegro, and C. Schumacher. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919 (Informational), August 2007.

[9] Moteiv Corporation. TMote Sky: Ultra low power IEEE 802.15.4 compliant wireless sensor module, June 2006.

[10] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 641–648, 2006.

[11] C. Paxson, M. Allman, J. Chu, and M. Sargent. Computing TCP's Retransmission Timer (RFC 6298), June 2011.

[12] Z. Shelby and C. Bormann. *6LoWPAN: The Wireless Embedded Internet.* Wiley Publishing, 2010.

[13] Z. Shelby, K. Hartke, and C. Bormann. Constrained application protocol (CoAP), May 2013.

[14] Texas Instruments. Chipcon products: CC2420 Datasheet: 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver, March 2013.

[15] T. Winter, P. Thubert, J. Hui, P. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. Technical Report 6550, RFC Editor, Fremont, CA, USA, Mar. 2012.

[16] Zolertia. Z1 low-power wireless sensor network module, March 2010.